A

PROJECT REPORT

ON­­

“ DOCTOR APPOINTMENT SYSTEM USING PYTHON ”

SUBMITTED

To

CENTRE FOR ONLINE LEARNING

Dr. D.Y.PATIL VIDYAPEETH, PUNE

­­­­­­­



IN PARTIAL FULFILMENT OF THE DEGREE OF

MASTER OF BUSINESS ADMINISTRATION

BY

**LEARNER NAME: REKHA KUMBHARKAR**

**PRN: 221117033**

**BATCH 2022-2024**



**Dr. D.Y. Patil Vidyapeeth’s**

**CENTRE FOR ONLINE LEARNING,**

**Sant TukaramNagar,Pune.**

CERTIFICATE

This is to certify that Mr./Ms. Rekha Santosh Kumbharkar

PRN - 221117033

has completed his/her internship at LTIMINDTREE

starting from 26 sept to 26 dec.

His / Her project work was a part of the MBA (ONLINE LEARNING)

The project is on Doctor Appointment System using python

Which includes research as well as industry practices. He/She was very sincere and committed in all tasks.

Course Coordinator

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date -

COMPANY LETTER

**(TO BE PROVIDED BY THE COMPANY WHERE THE PROJECT WILL BE CARRIED OUT)**

**To whomsoever it may concern**

This is to certify that Mr./Ms. Rekha Santosh Kumbharkar

PRN - 221117033

 has completed his/her internship at \_LTIMINDTREE

 starting from 26 Sep to 26 Dec.

His / Her project work was a part of the MBA (ONLINE LEARNING)

The project is on Doctor Appointment System Using Python

Which includes research as well as industry practices. He/ She was very sincere and committed in all tasks.

Signature & Seal of Industry Guide

**DECLARATION BY LEARNER**

This is to declare that I have carried out this project work myself in part fulfilment of the M.B.A Program of Centre for Online Learning of Dr.D.Y.Patil Vidyapeeth’s, Pune – 411018

The work is original, has not been copied from anywhere else, and has not been submitted to any other University / Institute for an award of any degree / diploma.

Date: - Signature: -

Place: Name:

**ACKNOWLEDGEMENT**

**(TO BE GIVEN BY THE LEARNER)**

**Rekha Kumbharkar,  
Pune 411041,**[**rekhakumbharkar16@gmail.com**](mailto:rekhakumbharkar16@gmail.com) **9172900795**

**10/08/2024**

**Mr. Bijal Thaker ,  
Asst. Professor ,  
Dy Patil University, Pune**

Dear **Bijal Thaker,**

I am writing this letter to express my sincere gratitude for your invaluable guidance, support, and encouragement throughout the completion of my project titled **Doctor Appointment System Using Python**. Your expertise and mentorship have been instrumental in shaping the success of this project.

Your [specific contributions, e.g., providing resources, sharing knowledge, offering feedback] were crucial in helping me develop a deeper understanding of the subject matter and overcome challenges encountered during the project. Your constructive criticism and suggestions have significantly enhanced the quality of my work.

I am truly fortunate to have had the opportunity to work under your supervision. Your belief in my abilities has motivated me to strive for excellence.

Thank you once again for your unwavering support.

Sincerely, Rekha.

**Table of content**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Item | Page No |
| 1 | Executive Summary | 1 |
| 2 | Chapter 1: Introduction (Company Profile & General Introduction of Topic)  & Objective, Scope, and Purpose of Study | 5 |
| 3 | Chapter 2: Literature Review | 7 |
| 4 | Chapter 3: Research Methodology | 9 |
| 5 | Chapter 4: Data Analysis | 10 |
| 6 | Chapter 5: Findings, suggestions, recommendation | 28 |
| 7 | Chapter 6: Conclusion | 30 |
| 8 | Bibliography (Books, Journals, research work) | 31 |
| 9 | Reference (Website, company paper) | 33 |
| 10 | Annexure (A to C) | 34 |
| 11 | A.Questionnaire | 34 |
| 12 | 1. Scope for future study | 35 |
| 13 | 1. Photograph, Drawings | 36 |

**Executive Summary**

**Problem:**

The traditional method of booking doctor appointments often involves long wait times, manual scheduling, and inefficient resource management, leading to patient dissatisfaction and operational challenges for healthcare providers.

**Solution:**

The Doctor Appointment System is a web-based application designed to streamline the appointment booking process for both patients and healthcare providers. By leveraging technology, the system aims to enhance patient experience, optimize clinic operations, and improve overall healthcare delivery.

**Key Features:**

* **Online Appointment Booking:** Patients can easily book, modify, or cancel appointments through a user-friendly interface.
* **Doctor Availability:** The system displays real-time doctor availability, allowing patients to choose preferred appointment times.
* **Patient Management:** Comprehensive patient records, including medical history and contact information, can be securely managed.
* **Appointment Reminders:** Automated reminders are sent to patients to reduce missed appointments.
* **Clinic Management:** Doctors can efficiently manage their schedules, view patient information, and generate appointment reports.

**Chapter 1: Introduction (Company Profile & General Introduction of Topic)**

**INTRODUCTION**

* **The Rise of Digital Healthcare**

The healthcare industry is undergoing a digital transformation, with technology playing an increasingly pivotal role in delivering efficient and accessible care. One area ripe for innovation is appointment scheduling. Traditional methods often involve time-consuming phone calls, manual scheduling, and potential overbooking.

**Challenges with Traditional Appointment Scheduling**

1. **Time-Consuming**: Patients often spend a considerable amount of time on phone calls to book appointments. This process can be frustrating, especially if the lines are busy or if they have to call multiple times.
2. **Manual Scheduling**: Hospitals and clinics typically rely on staff to manually schedule appointments. This can lead to errors, double bookings, and inefficiencies.
3. **Overbooking**: Without a centralized system, there’s a risk of overbooking, which can result in long wait times and a poor patient experience.
4. **Accessibility Issues**: Patients with mobility issues or those living in remote areas may find it challenging to visit the hospital just to book an appointment.

**Digital Solutions for Appointment Scheduling**

1. **Online Booking Systems**: These systems allow patients to book appointments online at their convenience. They can choose their preferred doctor, select a suitable time slot, and receive instant confirmation.
2. **Automated Reminders**: Digital systems can send automated reminders to patients via SMS or email, reducing the likelihood of missed appointments.
3. **Real-Time Availability**: Patients can see real-time availability of doctors, making it easier to find and book appointments without the risk of overbooking.
4. **Patient Portals**: These portals provide a centralized platform where patients can manage their appointments, view their medical history, and communicate with healthcare providers.

**Benefits of Digital Healthcare**

1. **Improved Efficiency**: Digital systems streamline the appointment scheduling process, reducing the workload on hospital staff and minimizing errors.
2. **Enhanced Patient Experience**: Patients enjoy a more convenient and user-friendly experience, with the ability to book appointments from anywhere at any time.
3. **Better Resource Management**: Hospitals can better manage their resources, ensuring that doctors’ schedules are optimized and that there are no overlaps or gaps in appointments.
4. **Data Analytics**: Digital systems can collect and analyze data on appointment bookings, helping healthcare providers identify trends, predict demand, and make informed decisions.

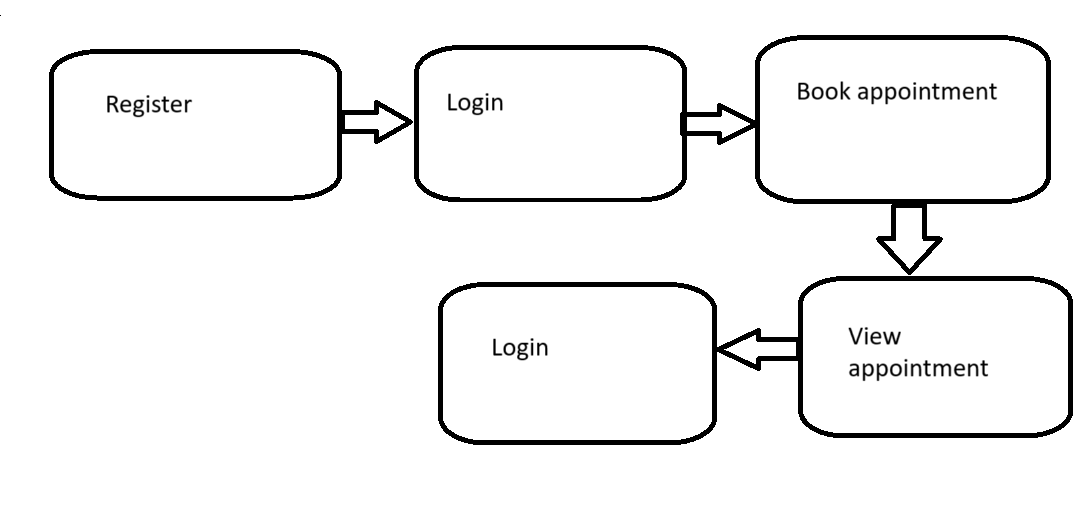
**Future Trends in Digital Healthcare**

1. **Telemedicine**: The rise of telemedicine allows patients to consult with doctors remotely, further enhancing accessibility and convenience.
2. **AI and Machine Learning**: These technologies can be used to predict patient no-shows, optimize scheduling, and provide personalized recommendations for appointment times.
3. **Integration with Wearable Devices**: Wearable health devices can provide real-time health data to doctors, enabling more proactive and personalized care.
4. **Blockchain for Data Security**: Blockchain technology can ensure the security and privacy of patient data, making digital healthcare systems more trustworthy.

* **Leveraging Python and Flask for a Solution**

To address these challenges, a doctor appointment system can be developed using Python, a versatile and powerful programming language, and Flask, a lightweight and flexible web framework. This combination provides a robust foundation for building a scalable and user-friendly application.

The Medicare industry is one of the rapidly growing industries in the world, because new diseases are identified very fast. People tend to visit hospital when they got a disease. To visit a hospital, people need to book appointment before visiting. In the previous years, booking appointment is a difficult task. People have to book appointment by dialling to the hospital or visiting the hospital itself. Dialling the hospital need some research for getting the hospital contact number. And sometimes, the hospital lines will be busy, So on continuous dialling the person can get the hospital line. In the hospital point of view, the appointment booing is done manually, it needs some human source. And also scheduling the appointment in a busy hospital manually is a difficult process. And also, the ringing phones may also disturb the patients admitted in that hospital. Visiting the hospital has its own disadvantage. If the patient is near to that hospital, it is fine to visit hospital for booking appointment, or else the patient has to travel a long distance to book the appointment and also, he/she have to travel the same distance on the appointment day. And also, the patients have to wait in the long queue for a long time which is an irritating process. In the hospital point of view visiting the hospital also needs some human source. Also, it makes the crowd in the hospital, so that some new problem may arise. To conclude this entire problem an online medical appointment booking system comes into act. Using this system people can book their appointments from anywhere and anytime. They can select any doctors and book their appointments at their time of comfort. They can view the doctor’s complete profile, ratings and reviews and if the patient is satisfied with the doctor, then he/she can book the appointment.

****

**COMPANY PROFILE**

LTIMINDTREE is a technology-driven healthcare and multinational solutions provider committed to enhancing patient care and operational efficiency with IT solutions. With a focus on leveraging innovative digital platforms, we aim to bridge the gap between patients and healthcare providers.

**The Challenge**

The healthcare industry faces significant challenges in managing patient appointments. Traditional manual systems often result in long wait times, overbooking, and inefficient resource allocation. These inefficiencies lead to patient dissatisfaction, increased operational costs, and a suboptimal patient experience.

**The Solution**

To address these challenges, we have developed a robust and user-friendly Doctor Appointment System. This system is designed to revolutionize the way appointments are scheduled and managed, benefiting both patients and healthcare providers.

**Objective, Scope, and Purpose of Study**

1. **Objective**

The primary objective of a doctor appointment system is to **streamline the appointment scheduling process** for both patients and healthcare providers. By automating and centralizing appointment management, the system aims to:

For Patient:-

1. Convince

Patients can easily book, reschedule, or cancel appointments online

without needing to call the clinic. This saves time and effort.

1. Accessibility

The system is available 24/7, allowing patients to schedule appointments

at their convenience, even outside of regular office hours.

1. Efficiency

Automated reminders and notifications help patients remember their appointments,

reducing the likelihood of missed appointments.

1. Information

Patients can access important information such as doctor availability, appointment history, and any pre-appointment instructions, ensuring they are well-prepared.

For Healthcare:-

1. Efficiency

Automating the scheduling process reduces the administrative burden on staff,

allowing them to focus on patient care.

1. Organization

The system helps in managing appointments, reducing overbooking or double-booking, and ensuring a smooth flow of patients.

1. Productivity

By minimizing no-shows and optimizing appointment slots,

healthcare providers can see more patients and improve their productivity.

1. Insights

Patient Demographics: Understanding the age, gender, and health conditions of patients can help tailor the appointment system to meet their specific needs.

Peak Times: Analyzing the times when appointments are most frequently booked can assist in resource allocation and staff scheduling.

No-Show Rates: Tracking the number of missed appointments can lead to better follow-up systems and overbooking strategies to minimize lost time.

Appointment Lead Time: The average time between booking and the actual appointment can provide insights into patient urgency and help optimize scheduling.

Patient Feedback: Collecting and analysing feedback post-appointment can reveal areas for improvement in the appointment process and overall patient care.

Technology Integration: The use of online booking systems, reminders via SMS or email, and digital check-ins can enhance the patient experience and reduce administrative workload.

Health Outcomes: Monitoring the outcomes of appointments, such as referral rates, prescription rates, and follow-up appointments, can indicate the effectiveness of consultations.­­­

1. **Scope**

The scope of a doctor appointment system using Python can vary based on the

project's complexity and specific requirements. However, a basic system should

include the following core functionalities:

**Core Functionality**

Core Functionality:-

User Management: Patient Registration and login

Technical Considerations**:-**

Python:- Flask Framework for website design

The Flask framework is a lightweight and flexible choice for web development.  
 It allows for rapid development and easy integration with other Python libraries.

Database:- SQLite  
 SQLite is a serverless database that is simple to set up and use.  
 It’s ideal for smaller projects and can handle moderate traffic with ease.

1. **Purpose of Study**

The purpose of this study is to develop a robust and user-friendly doctor appointment system using Python.

This system aims to address the inefficiencies and challenges associated with traditional appointment scheduling methods by providing a digital platform that streamlines the process for both patients and healthcare providers.

1. Improve patient experience
2. Enhance clinic efficiency
3. Leverage Technology

**Chapter 2: Literature Review**

**Purpose of the Literature Review**

A literature review for a doctor appointment system will help you:

* Understand the existing research: Identify previous studies, systems, or models related to appointment scheduling.
* Identify research gaps: Discover areas where there is a lack of research or opportunities for improvement.
* Inform system design: Gain insights into successful features, user preferences, and potential challenges.
* Support project justification: Demonstrate the need for your system by highlighting the limitations of current approaches.

**Key Areas to Explore**

1. Appointment Scheduling Systems:
   * Review existing appointment scheduling systems in healthcare, focusing on their strengths, weaknesses, and user experiences.
   * Analyse the impact of these systems on patient satisfaction, clinic efficiency, and resource utilization.
2. Online Appointment Booking:
   * Explore the benefits and challenges of online appointment booking for patients and healthcare providers.
   * Investigate factors influencing patient preferences for online booking.
3. Patient Satisfaction:
   * Examine studies on factors affecting patient satisfaction in healthcare settings.
   * Identify how appointment scheduling contributes to overall patient satisfaction.
4. Clinic Efficiency:
   * Review research on appointment scheduling optimization techniques.
   * Analyse the impact of appointment scheduling on clinic operations and resource allocation.
5. Technology Adoption:
   * Explore factors influencing the adoption of healthcare IT systems.
   * Identify strategies for successful implementation of appointment scheduling systems.

**Potential Research Questions**

* How have existing appointment scheduling systems impacted patient satisfaction and clinic efficiency?
* What are the key factors influencing the success of online appointment booking systems in healthcare?
* How can technology be used to optimize appointment scheduling and reduce no-show rates?
* What are the challenges and opportunities associated with integrating appointment scheduling systems with electronic health records (EHRs)?  
    
    
  Despite the increasing adoption of Web-based appointment systems, their potential benefits are yet to be systematically studied. The purpose of this review was to examine the current body of literature about Web-based medical appointment systems, specifically in regard to their potential benefits to patients and providers. We also want to identify the most effective services or components of them and explore the benefits and barriers of implementation. It is not the intention of this work to review the literature regarding fundamental theories of medical scheduling or system design,   
  There are two modes of Web-based appointment systems, asynchronous and real-time. In the asynchronous mode, appointments are requested through emails or electronic forms on providers’ website, and then manually processed by schedulers. In the real-time mode, patients can directly interact with providers’ scheduling management systems. Although the asynchronous Web-based appointment systems also use the Internet as a medium, they basically replicate the process of telephone-based appointment scheduling . Under the asynchronous mode, if an appointment is requested outside of a provider’s business hours, it will not be processed until schedulers return to work. Normally, Web-based appointment requests are put in the same queue as phone-call appointments, and are thus limited by the backlog of phone calls in the queue.

**Chapter 3: Research Methodology**

**Research Design**

Given the nature of the project, a **descriptive research design** would be suitable. This approach will involve gathering data to describe the current state of appointment scheduling systems, identifying problems, and exploring potential solutions.

**Research Objectives**

Before diving into the methodology, it's crucial to define clear research objectives. These could include:

* Identifying the pain points of the current appointment booking process
* Evaluating the effectiveness of existing appointment systems
* Exploring the potential benefits of a new appointment system
* Determining the required functionalities of a new system

**Flowchart**

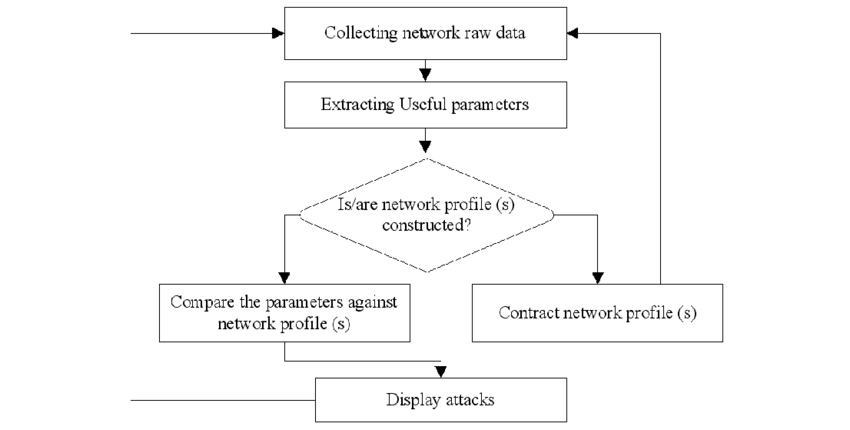


Fig.1 Flowchart for Research Methodology

**Chapter 4: Data Analysis**

**Understanding the Data**

Before diving into specific analysis techniques, it's crucial to understand the kind of data you'll be working with. A typical doctor appointment system might collect data on:

* **Patient demographics:** User name, password for login if new patient then register first.
* **Appointment details:** Day, Date, time, doctor id.

**Appointment Scheduling and Utilization**

* **Appointment Distribution:** Analyze appointment frequency by day of the week, time of day, and doctor. Identify peak and off-peak times.
* **Appointment Duration:** Calculate average appointment duration and identify outliers.
* **Doctor Utilization:** Evaluate doctor schedules for optimal utilization and potential overbooking or underbooking.
* **Room Utilization:** Analyze room occupancy to optimize room allocation.

**Patient Behavior**

* **Appointment Cancellations and No-Shows:** Calculate cancellation and no-show rates, identify patterns, and analyze their impact.
* **Patient Wait Times:** Measure average wait times and identify factors affecting wait times.
* **Appointment Frequency:** Analyze patient appointment history to identify repeat patients and appointment patterns.

**System Performance**

* **Booking Success Rate:** Calculate the percentage of successful appointment bookings.
* **System Response Time:** Measure the time taken to complete various system operations.
* **Error Analysis:** Identify and analyze system errors or exceptions.

**Data Analysis Techniques**

Depending on your data and goals, you can use various techniques:

* **Descriptive Statistics:** Calculate measures of central tendency (mean, median, mode) and dispersion (standard deviation, variance) to summarize data.
* **Frequency Distributions:** Analyze the distribution of data to identify patterns and trends.
* **Correlation Analysis:** Identify relationships between variables (e.g., appointment duration and doctor specialization).
* **Data Visualization:** Create graphs, charts, and dashboards to visually represent data insights.

**Tools and Software**

* **Python:** Use libraries like Pandas, NumPy, and Matplotlib for data manipulation and visualization.
* **SQLAlchemy:** For database interactions.
* **Flask:** For web application development.

**Description of Project and their tools: -**

The Tools and Software we used I mentioned above

**A. Python: -**   
Python is a high-level programming language, which is easy to program and understand and similar to how we speak in English.

Python is used in various applications like development, data processing, data analysis, artificial intelligence, machine learning, deep learning and scientific computing. It is a versatile language that can be used for both simple as well as complex projects.

Features of Python: -  
1. Simple Syntax and structure   
2. Easy to learn  
3. Library (For every function)  
4. High level language with open source.  
  
The main and Important feature of python is   
**Interpreted: -**

A diagram of a code

Description automatically generatedIn interpreted languages, the step of first converting the main code into a binary code does not take place. The code is then converted to machine code during runtime, which makes the language easy to debug and portable, as seen in the flowchart below.

5. It’s a procedure oriented as well as object-oriented programming language.  
6. Having wide range of Built-in Functions.  
7. Supports Exception Handling and Supports GUI

- In this project, we used the python language because of its simplicity there is no other option we used for database and web development separated within one code or one folder we created the whole project

- For doing this simultaneously we used to prefer ‘FLASK framework’

**B. Flask Framework: -**

Flask is a web framework, it’s a Python module that lets you develop web applications easily. It’s has a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features.

It does have many cool features like URL routing, template engine. It is a WSGI web app framework.

**What is web framework?**

A Web Application Framework or a simply a Web Framework represents a collection of libraries and modules that enable web application developers to write applications without worrying about low-level details such as protocol, thread management, and so on

**What is Flask?**

Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine. Both are Pocco projects.

**WSGI**

The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications.

**Werkzeug**

Werkzeug is a WSGI toolkit that implements requests, response objects, and utility functions. This enables a web frame to be built on it. The Flask framework uses Werkzeg as one of its bases.

**jinja2**

jinja2 is a popular template engine for Python.A web template system combines a template with a specific data source to render a dynamic web page.

This allows you to pass Python variables into HTML templates like this:

|  |
| --- |
| <html>  <head>  <title>{{ title }}</title>  </head>  <body>  <h1>Hello {{ username }}</h1>  </body> </html> |

**Microframework**

Flask is often referred to as a microframework. It is designed to keep the core of the application simple and scalable.

Instead of an abstraction layer for database support, Flask supports extensions to add such capabilities to the application.

**Why is Flask a good web framework choice?**

Unlike the Django framework, Flask is very Pythonic. It’s easy to get started with Flask, because it doesn’t have a huge learning curve.

On top of that it’s very explicit, which increases readability. To create the “Hello World” app, you only need a few lines of code.

This is a boilerplate code example.

|  |
| --- |
| from flask import Flask app = Flask(\_\_name\_\_)  @app.route('/') def hello\_world():  return 'Hello World!'  if \_\_name\_\_ == '\_\_main\_\_':  app.run() |

If you want to develop on your local computer, you can do so easily. Save this program as server.py and run it with python server.py.

|  |
| --- |
| $ python server.py  \* Serving Flask app "hello"  \* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit) |

It then starts a web server which is available only on your computer. In a web browser open localhost on port 5000 (the URL) and you’ll see “Hello World” show up.  
To host and develop online, you can use [PythonAnywhere](https://www.pythonanywhere.com/?affiliate_id=00535ced)

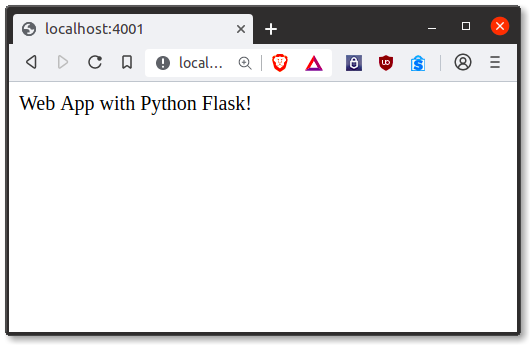
It’s a microframework, but that doesn’t mean your whole app should be inside one single Python file. You can and should use many files for larger programs, to handle complexity.

Micro means that the Flask framework is simple but extensible. You may all the decisions: which database to use, do you want an ORM etc, Flask doesn’t decide for you.

Flask is one of the most popular web frameworks, meaning it’s up-to-date and modern. You can easily extend its functionality. You can scale it up for complex applications.

- After doing the app routes for run the program once activate the scripts using the following command:

cd path/to/doctor\_appointement   
python main.py

To activate the python database   
python -c “from database import init\_db; init\_db()”  
  
one link will their ctrl and click <http://127.0.0.1:5000/> (like this one link is there)  
  


**C. SQLite: -**

SQLite is a C library that provides a lightweight disk-based database that doesn’t require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It’s also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

<https://www.sqlite.org/>   
<https://www.w3schools.com/sql/>   
  
  
**D.SQLAlchemy**

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

The major sections of this tutorial are as follows:

* [Establishing Connectivity - the Engine](https://docs.sqlalchemy.org/en/20/tutorial/engine.html#tutorial-engine) - all SQLAlchemy applications start with an [Engine](https://docs.sqlalchemy.org/en/20/core/connections.html#sqlalchemy.engine.Engine) object; here’s how to create one.
* [Working with Transactions and the DBAPI](https://docs.sqlalchemy.org/en/20/tutorial/dbapi_transactions.html#tutorial-working-with-transactions) - the usage API of the [Engine](https://docs.sqlalchemy.org/en/20/core/connections.html#sqlalchemy.engine.Engine) and its related objects [Connection](https://docs.sqlalchemy.org/en/20/core/connections.html#sqlalchemy.engine.Connection) and [Result](https://docs.sqlalchemy.org/en/20/core/connections.html#sqlalchemy.engine.Result) are presented here. This content is Core-centric however ORM users will want to be familiar with at least the [Result](https://docs.sqlalchemy.org/en/20/core/connections.html#sqlalchemy.engine.Result) object.
* [Working with Database Metadata](https://docs.sqlalchemy.org/en/20/tutorial/metadata.html#tutorial-working-with-metadata) - SQLAlchemy’s SQL abstractions as well as the ORM rely upon a system of defining database schema constructs as Python objects. This section introduces how to do that from both a Core and an ORM perspective.
* [Working with Data](https://docs.sqlalchemy.org/en/20/tutorial/data.html#tutorial-working-with-data) - here we learn how to create, select, update and delete data in the database. The so-called [CRUD](https://docs.sqlalchemy.org/en/20/glossary.html#term-CRUD) operations here are given in terms of SQLAlchemy Core with links out towards their ORM counterparts. The SELECT operation that is introduced in detail at [Using SELECT Statements](https://docs.sqlalchemy.org/en/20/tutorial/data_select.html#tutorial-selecting-data) applies equally well to Core and ORM.
* [Data Manipulation with the ORM](https://docs.sqlalchemy.org/en/20/tutorial/orm_data_manipulation.html#tutorial-orm-data-manipulation) covers the persistence framework of the ORM; basically the ORM-centric ways to insert, update and delete, as well as how to handle transactions.
* [Working with ORM Related Objects](https://docs.sqlalchemy.org/en/20/tutorial/orm_related_objects.html#tutorial-orm-related-objects) introduces the concept of the [relationship()](https://docs.sqlalchemy.org/en/20/orm/relationship_api.html#sqlalchemy.orm.relationship) construct and provides a brief overview of how it’s used, with links to deeper documentation.
* [Further Reading](https://docs.sqlalchemy.org/en/20/tutorial/further_reading.html#tutorial-further-reading) lists a series of major top-level documentation sections which fully document the concepts introduced in this tutorial.

**E. Web Development: -**- In this already using flask it created the html code so we need not to add many for that but yes we for the design purpose we can use CSS and JS for the interactive page which will be future scope of this project.

**DATA FLOW OF THE PROJECT**: -

* Doctor appointment
  1. Main.py
  2. Views.py
  3. Models.py
  4. Database.py
  5. Book\_appointment.html
  6. View\_appointment.html
* Templates   
  1. Login.html  
  2. Register.html

A screenshot of a computer program

Description automatically generated

Main.py :-

from flask import Flask, render\_template, request, redirect, url\_for, session

from database import init\_db

from views import register, login, dashboard, book\_appointment, view\_appointments

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

app.add\_url\_rule('/', 'index', view\_func=lambda: redirect(url\_for('login')))

app.add\_url\_rule('/register', 'register', view\_func=register, methods=['GET', 'POST'])

app.add\_url\_rule('/login', 'login', view\_func=login, methods=['GET', 'POST'])

app.add\_url\_rule('/dashboard', 'dashboard', view\_func=dashboard)

app.add\_url\_rule('/book\_appointment', 'book\_appointment', view\_func=book\_appointment, methods=['GET', 'POST'])

app.add\_url\_rule('/view\_appointments', 'view\_appointments', view\_func=view\_appointments)

if \_\_name\_\_ == '\_\_main\_\_':

    init\_db()

    app.run(debug=True)

Description: -

This [main.py](http://main.py) file is the entry point for a Flask web application. Below is a detailed explanation of what each part of the code does:

1. Importing Necessary Modulesfrom flask import Flask, redirect, url\_for

from database import init\_db

from views import register, login, dashboard, book\_appointment, view\_appointmentsFlask: This is the main class used to create a Flask application.redirect: This function is used to redirect the user to a different URL.url\_for: This function is used to build a URL for a given function name. It’s useful for creating dynamic links in the application.init\_db: This function is imported from [database.py](http://database.py) and is responsible for initializing the database by creating necessary tables.views (register, login, dashboard, book\_appointment, view\_appointments): These are functions defined in the [views.py](http://views.py) file that handle the logic for different parts of the web application.

2. Creating the Flask Appapp = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'app = Flask(name): This line creates an instance of the Flask class. The \_\_name\_\_ argument tells Flask to set up paths correctly and find other components relative to the location of [main.py.app](http://main.py.app).secret\_key: This is a secret key required for session management and security-related features such as CSRF protection. It's essential for securely signing the session cookie. Replace 'your\_secret\_key' with a strong secret key.

3. Setting Up URL Routesapp.add\_url\_rule('/', 'index', view\_func=lambda: redirect(url\_for('login')))

app.add\_url\_rule('/register', 'register', view\_func=register, methods=['GET', 'POST'])

app.add\_url\_rule('/login', 'login', view\_func=login, methods=['GET', 'POST'])

app.add\_url\_rule('/dashboard', 'dashboard', view\_func=dashboard)

app.add\_url\_rule('/book\_appointment', 'book\_appointment', view\_func=book\_appointment, methods=['GET', 'POST'])

app.add\_url\_rule('/view\_appointments', 'view\_appointments', view\_func=view\_appointments)app.add\_url\_rule: This method is used to bind a URL to a function (view) in the application.'/': The root URL of the application. When users visit the root URL, they are redirected to the login page.'/register': URL for the user registration page, handled by the register view.'/login': URL for the login page, handled by the login view.'/dashboard': URL for the dashboard page, handled by the dashboard view.'/book\_appointment': URL for booking an appointment, handled by the book\_appointment view.'/view\_appointments': URL for viewing booked appointments, handled by the view\_appointments view.view\_func: This specifies which function should be called when the route is accessed.methods=['GET', 'POST']: This specifies that the route can handle both GET and POST HTTP methods. GET is typically used for fetching data or rendering a page, while POST is used for submitting form data.

4. Running the Applicationif \_\_name\_\_ == '\_\_main\_\_':

 init\_db()

[app.run](http://app.run)(debug=True)if name == 'main': This block ensures that the application runs only if the script is executed directly (not imported as a module).init\_db(): This function initializes the database (creates tables if they don’t already exist).[app.run](http://app.run)(debug=True): This starts the Flask web server. The debug=True argument enables debug mode, which provides useful error messages and automatically reloads the server when code changes are made.SummaryThe application is a basic Flask app that supports user registration, login, and managing appointments.The app.add\_url\_rule function maps URLs to view functions, determining how the app should respond to different web requests.The init\_db function initializes the database, setting up the necessary tables.The [app.run](http://app.run)(debug=True) starts the server in debug mode, allowing you to see errors and automatically restart the server when code changes.

Views.py

from flask import render\_template, request, redirect, url\_for, session

from database import db\_session

from models import User, Appointment

def register():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        user = User(username=username, password=password)

        db\_session.add(user)

        db\_session.commit()

        return redirect(url\_for('login'))

    return render\_template('register.html')

def login():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        user = User.query.filter\_by(username=username, password=password).first()

        if user:

            session['user\_id'] = [user.id](http://user.id/)

            return redirect(url\_for('dashboard'))

    return render\_template('login.html')

def dashboard():

    if 'user\_id' not in session:

        return redirect(url\_for('login'))

    return render\_template('dashboard.html')

def book\_appointment():

    if request.method == 'POST':

        doctor\_id = request.form['doctor\_id']

        appointment\_time = request.form['appointment\_time']

        appointment = Appointment(user\_id=session['user\_id'], doctor\_id=doctor\_id, appointment\_time=appointment\_time)

        db\_session.add(appointment)

        db\_session.commit()

        return redirect(url\_for('dashboard'))

    return render\_template('book\_appointment.html')

def view\_appointments():

    appointments = Appointment.query.filter\_by(user\_id=session['user\_id']).all()

    return render\_template('view\_appointments.html', appointments=appointments)

Description: -

Importing Dependencies: Flask: The primary web framework used to create and run the web application. Redirect and url\_for: Utilities from Flask used for URL redirection and building dynamic URLs.init\_db: A custom function imported from [database.py](http://database.py) to initialize the database.

Views: Functions (like register, login, etc.) imported from [views.py](http://views.py) that handle specific web pages or "views" in the app. Flask Application Instance: app = Flask(name): This line creates the Flask application instance. The \_\_name\_\_ argument helps Flask determine the root path of the [application.app](http://application.app).secret\_key = 'your\_secret\_key': This sets the secret key for the application, which is essential for securely signing the session cookie. The secret key is crucial for session management, protecting against certain security vulnerabilities.URL Routing: The app.add\_url\_rule method binds specific URLs to functions (views) that handle the logic for each web page./ (Root URL): When users visit the root URL, they are redirected to the login page using redirect(url\_for('login'))./register: Handles user registration. This route supports both GET (rendering the registration form) and POST (submitting the registration form) methods./login: Handles user login, also supporting both GET and POST methods./dashboard: After logging in, users are redirected to the dashboard, where they can see their options./book\_appointment: Allows users to book an appointment. This route also supports GET and POST methods./view\_appointments: Displays a list of the user's booked appointments. Database Initialization and Application Launch: if name == 'main': This ensures the code inside this block runs only when the script is executed directly, not when it's imported as a module.init\_db(): This function initializes the database by creating all necessary tables based on the models defined in the [application.app.run](http://application.app.run)(debug=True): Starts the Flask development server in debug mode. Debug mode provides detailed error messages and automatically reloads the application when code changes are detected.

Models.py

from sqlalchemy import Column, Integer, String, ForeignKey, DateTime

from sqlalchemy.orm import relationship

from database import Base

class User(Base):

    \_\_tablename\_\_ = 'users'

    id = Column(Integer, primary\_key=True)

    username = Column(String(50), unique=True)

    password = Column(String(50))

class Appointment(Base):

    \_\_tablename\_\_ = 'appointments'

    id = Column(Integer, primary\_key=True)

    user\_id = Column(Integer, ForeignKey('[users.id](http://users.id/)'))

    doctor\_id = Column(Integer, ForeignKey('[users.id](http://users.id/)'))

    appointment\_time = Column(DateTime)

    user = relationship("User", foreign\_keys=[user\_id])

    doctor = relationship("User", foreign\_keys=[doctor\_id])

Description:-

The [models.py](http://models.py) script defines the structure of the database tables using SQLAlchemy’s ORM (Object Relational Mapping). This script is crucial because it maps Python classes to database tables, allowing for easy data manipulation and querying.

1. Importing Required Modules :  
from sqlalchemy import Column, Integer, String, ForeignKey, DateTime

from sqlalchemy.orm import relationship

from database import BaseColumn, Integer, String, ForeignKey, DateTime:   
These are SQLAlchemy classes used to define the columns and data types in the database tables.relationship: This is used to define relationships between different tables (e.g., foreign key relationships).  
Base: This is the base class that was defined in [database.py](http://database.py) using declarative\_base().   
All models in this script inherit from this Base class, which ties them to the database.

2. User Modelclass User(Base):

    \_\_tablename\_\_ = 'users'

    id = Column(Integer, primary\_key=True)

    username = Column(String(50), unique=True)

password = Column(String(50))class User(Base): This defines a User class that maps to a table in the database named users.tablename = 'users': This specifies the name of the database [table.id](http://table.id) = Column(Integer, primary\_key=True): This defines an id column as an integer and sets it as the primary key for the table. The primary key uniquely identifies each record in the table.username = Column(String(50), unique=True): This defines a username column that can store strings up to 50 characters. The unique=True constraint ensures that no two users can have the same username.password = Column(String(50)): This defines a password column that can store strings up to 50 characters.  
  
3. Appointment Modelclass Appointment(Base):

    \_\_tablename\_\_ = 'appointments'

    id = Column(Integer, primary\_key=True)

user\_id = Column(Integer, ForeignKey('[users.id](http://users.id)'))

doctor\_id = Column(Integer, ForeignKey('[users.id](http://users.id)'))

    appointment\_time = Column(DateTime)

    user = relationship("User", foreign\_keys=[user\_id])

doctor = relationship("User", foreign\_keys=[doctor\_id])class Appointment(Base):   
This defines an Appointment class that maps to a table in the database named appointments.tablename = 'appointments':   
This specifies the name of the database [table.id](http://table.id) = Column(Integer, primary\_key=True): This defines an id column as an integer and sets it as the primary key for the table.user\_id = Column(Integer, ForeignKey('[users.id](http://users.id)')): This defines a user\_id column as an integer and sets it as a foreign key referencing the id column in the users table.   
This indicates which user booked the appointment.doctor\_id = Column(Integer, ForeignKey('[users.id](http://users.id)')): This defines a doctor\_id column as an integer and sets it as a foreign key referencing the id column in the users table.   
This indicates which doctor the appointment is with.appointment\_time = Column(DateTime): This defines an appointment\_time column to store the date and time of the appointment.user = relationship("User", foreign\_keys=[user\_id]): This sets up a relationship between the Appointment and User tables, linking the user\_id to a user.   
The foreign\_keys argument specifies which column in the Appointment table links to the User table.doctor = relationship("User", foreign\_keys=[doctor\_id]): Similarly, this sets up a relationship between the Appointment and User tables, linking the doctor\_id to a user (who is acting as a doctor in this context).

Database.py: -

from sqlalchemy import create\_engine  
from sqlalchemy.ext.declarative import declarative\_base  
from sqlalchemy.orm import scoped\_session, sessionmaker

DATABASE\_URL = 'sqlite:///doctor\_appointment.db'  # Database URL

engine = create\_engine(DATABASE\_URL, convert\_unicode=True)  
db\_session = scoped\_session(sessionmaker(autocommit=False, autoflush=False, bind=engine))

Base = declarative\_base()  
Base.query = db\_session.query\_property()

def init\_db():  
    import models  
    Base.metadata.create\_all(bind=engine)

Description: -

1. Importing Required Modulesfrom sqlalchemy import create\_engine, Column, Integer, String, ForeignKey, DateTime

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import scoped\_session, sessionmaker, relationshipsqlalchemy:

This is the main library used for database operations. It provides tools for working with relational databases in Python.create\_engine: This function creates a connection to the database.Column, Integer, String, ForeignKey, DateTime: These are SQLAlchemy classes used to define the columns and data types in the database tables.declarative\_base: This function provides a base class for creating models using SQLAlchemy's ORM (Object Relational Mapping) system.scoped\_session, sessionmaker: These are used to create a session for database transactions, allowing you to query the database and make changes.relationship: This function is used to define relationships between tables.

2. Creating the Database Engine engine = create\_engine('sqlite:///appointments.db')create\_engine:

This function initializes the connection to the database. Here, it creates an SQLite database named appointments.db.sqlite:///appointments.db: This is the database URL. SQLite is a lightweight database that stores data in a file. The triple slashes indicate the file path.

3. Creating a Scoped Sessiondb\_session = scoped\_session(sessionmaker(autocommit=False, autoflush=False, bind=engine))scoped\_session:

This creates a thread-safe session that can be used across different parts of the application.sessionmaker: This function creates a new session factory.autocommit=False: This setting disables automatic commit of transactions, giving you control over when changes are saved to the database.autoflush=False: This setting prevents SQLAlchemy from automatically flushing changes to the database until explicitly requested.bind=engine: This binds the session to the previously created database engine.

4. Defining the Base ClassBase = declarative\_base()  
Base.query = db\_session.query\_property()Base = declarative\_base():

This creates a base class that all database models will inherit from. This base class includes methods and attributes that allow the models to interact with the database.Base.query = db\_session.query\_property(): This adds a query property to the base class, allowing you to perform queries on models easily. For example, you can use Model.query to query data from the corresponding table.

5. Initializing the Databasedef init\_db():

    # Importing models to ensure they are registered properly on the metadata

    import models

    # Creating all tables

Base.metadata.create\_all(bind=engine)init\_db(): This function is responsible for initializing the database.import models: This imports the [models.py](http://models.py) file, which contains the table definitions. Importing it here ensures that all models are registered with SQLAlchemy before creating the tables.Base.metadata.create\_all(bind=engine):

This command creates all tables in the database based on the models defined in [models.py](http://models.py). If the tables already exist, this function does nothing.SummaryThe [database.py](http://database.py) script sets up the core database functionality for the web application. It creates a connection to an SQLite database, establishes a session for interacting with the database, and defines a base class for models. The init\_db() function is crucial for initializing the database by creating all the necessary tables based on the defined models. This setup allows the application to perform database operations like querying and inserting records in a structured and efficient manner.

Login.html  
  
<!doctype html>  
<html>  
  <head><title>Login</title></head>  
  <body>  
    <h2>Login</h2>  
    <form method="post">  
      Username: <input type="text" name="username"><br>  
      Password: <input type="password" name="password"><br>  
      <input type="submit" value="Login">  
    </form>  
  </body>  
</html>  
  
  
Register.html  
  
<!doctype html>  
<html>  
  <head><title>Register</title></head>  
  <body>  
    <h2>Register</h2>  
    <form method="post">  
      Username: <input type="text" name="username"><br>  
      Password: <input type="password" name="password"><br>  
      <input type="submit" value="Register">  
    </form>  
  </body>  
</html>  
  
  
Book Appointment.html

<!doctype html>  
<html>  
  <head><title>Book Appointment</title></head>  
  <body>  
    <h2>Book Appointment</h2>  
    <form method="post">  
      Doctor ID: <input type="text" name="doctor\_id"><br>  
      Appointment Time: <input type="datetime-local" name="appointment\_time"><br>  
      <input type="submit" value="Book">  
    </form>  
  </body>  
</html>

View Appointment.html

<!doctype html>  
<html>  
  <head><title>View Appointments</title></head>  
  <body>  
    <h2>Your Appointments</h2>  
    <ul>  
      {% for appointment in appointments %}  
      <li>{{ appointment.appointment\_time }} with Doctor ID: {{ appointment.doctor\_id }}</li>  
      {% endfor %}  
    </ul>  
  </body>  
</html>

Dashboard.html

<!-- templates/dashboard.html -->  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Dashboard</title>  
</head>  
<body>  
    <h1>Welcome to the Dashboard</h1>  
    <p>This is the dashboard page.</p>  
    <a href="{{ url\_for('book\_appointment') }}">Book an Appointment</a>  
    <a href="{{ url\_for('view\_appointments') }}">View Appointments</a>  
    <a href="{{ url\_for('logout') }}">Logout</a>  
</body>  
</html>

**Chapter 5: Findings, suggestions, recommendation**

**Findings :-**  
We have find that easy way to book appointment for patient with name and password and if patient is not created they register .

We can do better at frontend side.  
  
**Suggestions :-**

**Frontend :-**

Improving the frontend involves creating a user interface that is not only aesthetically pleasing but also functional and efficient. Consider the following aspects:

User Interface Design: Adopt a design thinking approach to create a layout that is intuitive and aligns with the user’s needs. Use color schemes and typography that are easy on the eyes and accessible to users with disabilities.

Interactivity: Implement interactive elements such as dropdowns, date pickers, and sliders to make the booking process engaging and straightforward.

Mobile Responsiveness: Ensure the website is fully responsive and provides a seamless experience on mobile devices, as many users will access the system on the go.

Interactivity: Implement interactive elements such as dropdowns, date pickers, and sliders to make the booking process engaging and straightforward.

Mobile Responsiveness: Ensure the website is fully responsive and provides a seamless experience on mobile devices, as many users will access the system on the go.

Enhanced Patient Information Collection:

Gathering more detailed information from patients can lead to personalized care and better service:

Medical History: Include fields for medical history, allergies, and current medications to help doctors prepare for the appointment.

Symptom Checker: Integrate a symptom checker that can guide patients to the appropriate specialist based on their inputs.

Privacy Compliance: Make sure that the system complies with privacy laws and regulations, ensuring that patient data is collected and stored securely.

Doctor Database Management:

A well-structured database for doctors is crucial for an efficient appointment system:

Specialization and Availability: Store information about doctors’ specializations, qualifications, and availability to match them with patient needs effectively.

Performance Tracking: Keep track of doctors’ performance through patient feedback and appointment outcomes to maintain high service quality.

Real-Time Alerts: Send real-time alerts for any changes in the appointment schedule, such as delays or cancellations.

Integration with Scheduling: Ensure the database is integrated with the scheduling system for real-time updates and availability checks.

Notification System Implementation:

A robust notification system is key to keeping patients informed and engaged:

Multi-Channel Notifications: Use multiple channels like email, SMS, and in-app notifications to reach patients according to their preferences.

Technical Implementation:

On the technical side, consider the following for implementation:

Frontend Technologies: Use advanced frontend technologies like Angular, React, or Vue.js for a dynamic and single-page application experience.

Backend Integration: Ensure the frontend communicates effectively with the backend through RESTful APIs or GraphQL.

Security Measures: Implement security best practices such as HTTPS, data encryption, and secure authentication methods like OAuth.

**Recommendation :-**

More work on fronted system and gathering some more information from user, create new database for doctor with specific Id and department.

Notification system for paitent to know they appointment status

**Chapter 6: Conclusion**

**The developed doctor appointment system effectively addresses the challenges associated with traditional manual appointment booking systems.**

In conclusion, doctor appointment systems have become an essential part of modern healthcare services, as they enable patients to schedule appointments with healthcare professionals easily and efficiently. By reducing waiting times, improving scheduling processes, and reducing administrative burdens, doctor appointment systems can help to improve the overall quality of care for patients.

Additionally, as healthcare systems continue to evolve, the development of effective doctor appointment systems will be critical to ensuring that patients receive the care they need in a timely and efficient manner. However, designing and implementing a website for reserving data and storing patient information is an essential component of modern healthcare. A well-designed website can improve healthcare outcomes, enhance patient satisfaction, and streamline healthcare processes.

It is crucial to prioritize user accessibility, security, and compliance with relevant data protection and privacy regulations to establish trust between patients and healthcare providers. With careful planning and implementation, a website for reserving data and storing patient information can transform the delivery of healthcare services. the implementation of a doctor appointment system online can bring numerous benefits for both patients and healthcare providers.

By offering a user-friendly interface, mobile compatibility, and customizable booking options, patients can easily schedule appointments from anywhere and at any time, without experiencing long wait times or the frustration of traditional appointment booking systems. Overall, the development and implementation of a doctor appointment system online can bring significant benefits for patients, healthcare providers, and the healthcare system as a whole. It can help to streamline operations, improve patient experiences, and ultimately, enhance the quality of care provided to patients

Recommendations for Future Works There are several suggestions for future works:

1. Integration with Telemedicine: As more patients turn to telemedicine for remote consultations, the integration of telemedicine into the doctor appointment system can provide a comprehensive and convenient solution for patients.  
2. Patient Feedback: Gathering feedback from patients can help healthcare providers identify areas for improvement and tailor the booking system to meet the specific needs and preferences of patients.  
3. Integration with Wearable Devices: Integrating wearable devices, such as fitness trackers or smartwatches, can provide healthcare providers with real-time data on patients' health metrics. By implementing these future works, healthcare providers can continue to improve the doctor appointment system providing patients with convenient, efficient, and personalized care.

Limitations

The researchers suffer from many limitations during the design and implementation system, as listed in the following:   
1. Technical Issues: Like any technological solution, doctor appointment systems may experience technical issues, such as system crashes, slow loading times, or software bugs.   
2. Internet Connectivity: Doctor appointment systems are dependent on stable and reliable internet connectivity.

**Bibliography (Books, Journals, research work)**

**Research Work :-**Research Work for a Doctor Appointment System

While your project may not require extensive academic research, understanding the existing landscape and challenges in appointment scheduling can inform your system's design and functionality. Here are some potential areas of research:

Areas of Focus

* Appointment scheduling algorithms: Explore different algorithms for optimizing appointment scheduling, such as those used in airline or hotel booking systems.
* Patient satisfaction: Research factors influencing patient satisfaction with appointment systems, such as wait times, ease of use, and appointment availability.
* System usability: Study user interface design principles for healthcare applications to ensure a user-friendly system.
* Security and privacy: Investigate best practices for protecting patient data in healthcare systems.
* Integration with electronic health records (EHRs): Explore how appointment systems can be integrated with EHRs to improve efficiency and data sharing.

Potential Research Sources

* Academic Journals:
  + Journal of Biomedical Informatics
  + International Journal of Medical Informatics
  + Computers in Biology and Medicine
* Conference Proceedings:
  + Proceedings of the AMIA Annual Symposium
  + International Conference on Information Systems (ICIS)
* Industry Reports:
  + Reports from healthcare IT consulting firms
  + Market research reports on healthcare software

**Reference (Website, company paper)**

**Websites :-**

Basic and advanced python concepts.  
<https://www.w3schools.com/python/>   
<https://pythongeeks.org/features-of-python/>

For Flask :- For flask and easy way to create application using python   
<https://pypi.org/project/Flask/>   
<https://pythonbasics.org/what-is-flask-python/#:~:text=Flask%20is%20a%20web%20framework,like%20url%20routing%2C%20template%20engine>.   
<https://docs.python.org/3/library/sqlite3.html>   
  
Tutorial :-  
<https://www.geeksforgeeks.org/flask-tutorial/> , <https://en.wikipedia.org/wiki/Flask_(web_framework)>   
<https://flask.palletsprojects.com/en/3.0.x/>   
<https://www.javatpoint.com/python-frameworks#:~:text=17.-,Web2Py,all%20the%20popular%20operating%20systems>.   
<https://www.simplilearn.com/python-frameworks-article>   
<https://docs.python.org/3/library/frameworks.html>   
<https://robocorp.com/docs/python>   
<https://www.w3schools.com/html/html_css.asp>   
<https://www.codecademy.com/catalog/language/html-css>   
  
**Book Reference:-**Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming  
Python Programming: An Introduction to Computer Science

**Project Reference:-**<https://www.sourcecodester.com/tags/doctors-appointment-system-source-code>  
<https://github.com/topics/doctor-appointment-booking?l=python>   
<https://www.codeproject.com/>

**Annexure (A to C)**

1. Questionnaire
2. Scope for future study
3. Photograph, Drawings

**Questionnaire**   
  
Target Audience: Patients

Section A: Demographic Information

* Age
* Gender
* Occupation
* Location (urban/rural)
* Education level
* Smartphone ownership
* Internet access

Section B: Current Appointment Booking Experience

* How often do you visit a doctor in a year?
* What is your preferred method of booking an appointment (phone, in-person, online)?
* How satisfied are you with the current appointment booking process? (Likert scale)
* How long does it typically take to book an appointment?
* What are the major challenges you face when booking an appointment?
* How important is it for you to be able to book an appointment online? (Likert scale)

Section C: Desired Features

* What features would you like to see in an online appointment booking system? (e.g., appointment reminders, cancellation options, doctor profiles)
* How important is it for you to be able to choose a specific doctor or time slot for your appointment? (Likert scale)
* Would you prefer to receive appointment confirmations via SMS, email, or both?
* How important is it for you to be able to view your appointment history online? (Likert scale)

Section D: System Usability

* How easy is it to use online platforms for booking appointments for other services (e.g., restaurants, hotels)? (Likert scale)
* What features would make an online appointment booking system user-friendly?

Section E: Additional Comments

* Any other suggestions or feedback about appointment booking systems?

Target Audience: Doctors

Section A: Demographic Information

* Age
* Gender
* Specialization
* Years of experience
* Number of patients seen per day

Section B: Current Appointment Scheduling Process

* How do you currently manage patient appointments (paper-based, digital, or a combination)?
* How satisfied are you with the current appointment scheduling process? (Likert scale)
* What are the major challenges you face in managing appointments?
* How much time do you spend on appointment scheduling tasks per day?
* How important is it for you to have a digital overview of your schedule? (Likert scale)

Section C: Desired Features

* What features would you like to see in a doctor appointment booking system? (e.g., appointment reminders, patient profiles, availability management)
* How important is it for you to be able to manage your schedule online? (Likert scale)
* Would you like to have the option to offer online consultations? (Yes/No)
* How important is it for you to have real-time patient information available during appointments? (Likert scale)

Section D: System Usability

* How easy is it to use current medical software or electronic health records? (Likert scale)
* What features would make a doctor appointment booking system user-friendly for you?

Section E: Additional Comments

* Any other suggestions or feedback about appointment booking systems?

|  |
| --- |
|  |

**Scope for future study**

The scope for future study in a doctor appointment system can be quite extensive. Here are some potential areas:

System Enhancements

* Integration with Electronic Health Records (EHRs): Seamlessly integrating the appointment system with EHRs to improve data flow and patient care.
* Online Consultations: Incorporating video conferencing or chat features for virtual consultations.
* Patient Self-Service: Expanding patient self-service options, such as appointment cancellations, rescheduling, and prescription refills.
* Mobile App Development: Creating a mobile application for enhanced accessibility and user experience.
* Artificial Intelligence (AI) Integration: Implementing AI-powered features like appointment scheduling optimization, patient triage, and chatbots.

Data Analysis and Utilization

* Predictive Analytics: Utilizing data to forecast appointment demand and optimize resource allocation.
* Patient Behavior Analysis: Analyzing patient data to identify trends and preferences, enabling personalized services.
* Performance Metrics: Developing key performance indicators (KPIs) to monitor system performance and identify areas for improvement.

Expansion of Services

* Group Appointments: Offering group appointments for specific medical conditions or preventive care.
* Home Healthcare Integration: Coordinating home healthcare services through the appointment system.
* Telemedicine Integration: Integrating telemedicine capabilities for remote consultations.

Research and Development

* User Experience (UX) Optimization: Conducting usability studies to improve the system's interface and user experience.
* Security Enhancements: Implementing advanced security measures to protect patient data.
* Interoperability: Developing standards for data exchange with other healthcare systems.

By focusing on these areas, future research can contribute to the evolution of doctor appointment systems and improve the overall healthcare experience for patients and providers.

**Photograph, Drawings**

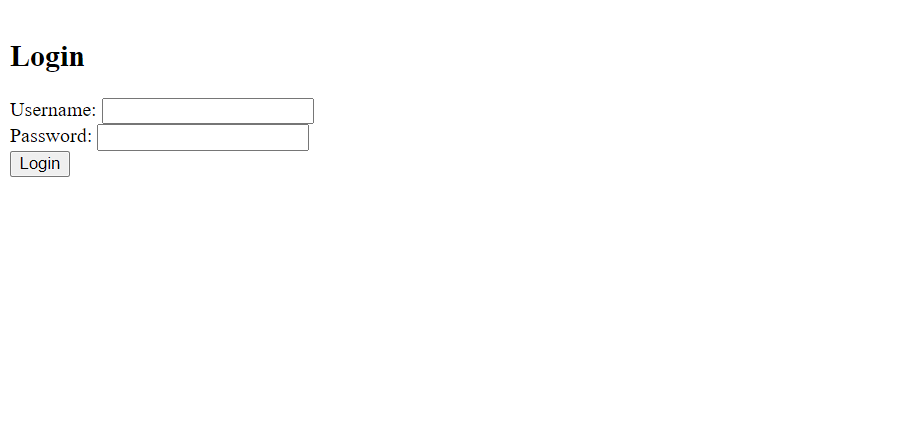


Fig 2. Login Page

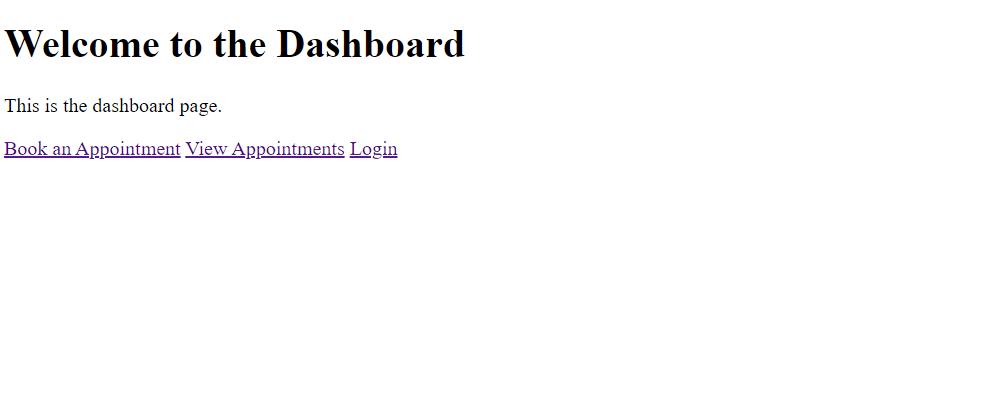


Fig 3. Dashboard Page

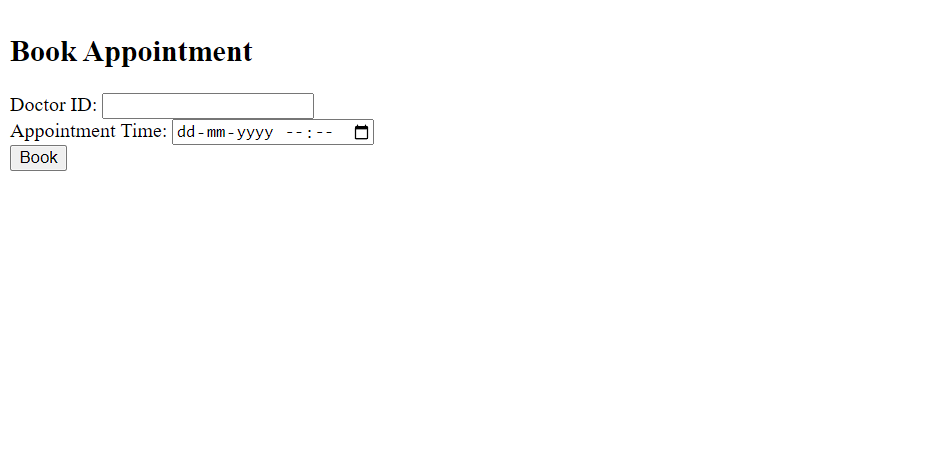


Fig 4. Book Appointment

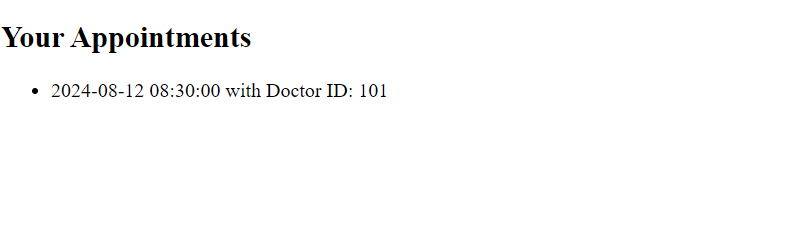


Fig 5. View Appointment

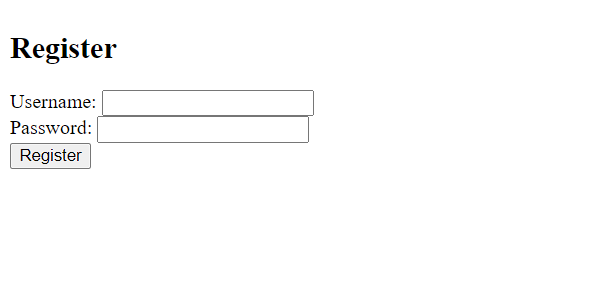


Fig 6. Register page